Um Ambiente para o Desenvolvimento baseado na Composição de Aplicações

Cláudia M.L. Werner
Flavio Araujo de Mattos

UFRJ/COPPE Sistemas
Cidade Universitária - CT - sala H-319
Caixa Postal 68511 - CEP 21945 - Rio de Janeiro - RJ

Resumo.

Este artigo apresenta um ambiente de suporte à reutilização baseado em uma tecnologia de composição. O ambiente fornece uma série de ferramentas especializadas que dão suporte à organização e manipulação de uma biblioteca de componentes de software reutilizáveis, facilitando o desenvolvimento de aplicações a partir da composição de software já existente.

O ambiente proposto combina as tecnologias de orientação a objetos e hipertexto, mostrando ser esta uma abordagem bastante adequada à implementação dos conceitos envolvidos em ambientes de reutilização por composição.

Abstract

This article presents a software environment for reuse by composition. The environment provides specialized tools which support the organization and manipulation of a reusable software library, promoting application development from the composition of existing software.

The proposed environment combines the object-oriented and hypertext tecnologies, showing the adequability of this approach to the implementation of the concepts involved in environments for software reuse by composition.

1. Introdução

A reutilização de software vem sendo discutida na literatura como uma possível estratégia para a redução dos custos de produção e tempo de desenvolvimento, a melhoria da qualidade e confiabilidade do software produzido [BIGG89]. Dentre as tecnologias atualmente disponíveis para o suporte ao desenvolvimento baseado em uma estratégia de reutilização de software, temos: geradores de aplicação, linguagens de quarta-geração, ambientes de suporte a bibliotecas de software, sistemas orientados a objetos, etc.

Biggerstaff e Richter [BIGG89] dividem estas tecnologias em dois grandes grupos: tecnologias de geração e tecnologias de composição. Neste artigo, apresentamos um ambiente de suporte à reutilização baseado em uma tecnologia de composição, denominado CAOS ("Composing Application-Objects' System") [WERN91]. O ambiente fornece uma série de ferramentas especializadas que dão suporte à organização e manipulação de uma biblioteca de componentes de software reutilizáveis, facilitando o desenvolvimento de aplicações a partir da composição de software já existente.

A idéia inicial do sistema era dar suporte ao desenvolvimento de aplicações no contexto científico, substituindo as atuais bibliotecas científicas por um sistema mais completo, capaz de dar apoio não somente às atividades de manipulação de bibliotecas mas, também, às demais atividades envolvidas na composição de aplicações (i.e. entendimento, modificação e composição de elementos de software). O projeto do ambiente, entretanto, tornou-se amplo o suficiente para torná-lo um ambiente adaptável a diferentes tipos de aplicação.

O Ambiente CAOS utiliza conceitos e técnicas de orientação a objetos. A programação orientada a objetos, através de seus mecanismos para organização e decomposição de sistemas em entidades encapsuladas (i.e. objetos e classes) e mecanismos para a modificação e composição incremental de software (i.e. herança,

generalização e ligação dinâmica) [TSIC88], provê facilidades básicas ao desenvolvimento de aplicações baseado na reutilização de componentes de software.

O ambiente utiliza, ainda, a tecnologia de hipertexto [CONK87], como estrutura organizacional e de navegação entre informações, e um esquema de classificação por facetas, tal como descrito por Prieto-Diaz [PRIE85], para recuperação de componentes reutilizáveis.

Na seção 2, o modelo de desenvolvimento de software proposto para o Ambiente CAOS é apresentado. Os requisitos do ambiente e de seus principais elementos fazem parte da seção 3. A implementação de um primeiro protótipo do sistema é descrita na seção 4. Finalmente, conclui-se o artigo na seção 5, identificando-se as características mais relevantes do ambiente proposto.

2. O Modelo de Desenvolvimento em CAOS

O modelo de desenvolvimento proposto para o Ambiente CAOS assume a existência de uma base de componentes reutilizáveis, munidos de sua documentação, e de um mecanismo de seleção capaz de localizar componentes relevantes à aplicação. Pode-se listar como etapas deste modelo [WERN92]:

- especificação da aplicação baseada na identificação de componentes básicos e na descrição das propriedades a serem satisfeitas pelos mesmos;
- seleção de componentes potencialmente úteis;
- compreensão e modificação de componentes selecionados;
- construção de novos componentes (caso seja necessária);
- composição da aplicação a partir dos componentes selecionados, adaptados ou construídos;
- observação do comportamento da aplicação através da sua execução, e
- continuação do desenvolvimento através da substituição e/ou evolução dos componentes.

Neste modelo, a ênfase é dada ao desenvolvimento contínuo, a partir da substituição ou evolução de componentes, não sendo gasto muito tempo na especificação da aplicação. A construção de uma aplicação é vista como o resultado de experiências na composição, especialização e generalização de componentes de um certo domínio de aplicação.

O sucesso deste modelo depende não somente da qualidade dos componentes disponíveis na base de componentes mas, também, de mecanismos de seleção suficientemente capazes de, a partir da descrição das propriedades dos componentes da aplicação feita na fase de especificação, localizar componentes potencialmente úteis.

3. Requisitos do Ambiente CAOS

O objetivo principal do Ambiente CAOS é prover um conjunto de ferramentas de suporte ao desenvolvimento baseado na composição de software. Para isto, o ambiente inclui os seguintes elementos [WERN92]:

- uma base de componentes de software;
- um gerente de componentes;
- um navegador;
- um acoplador de rotinas externas;
- um gerente de composição de aplicações;
- uma linguagem de composição de aplicações, e
- um monitor.

As sub-seções a seguir descrevem os requisitos de cada um desses elementos.

3.1. A Base de Componentes de Software e o Gerente de Componentes

A iniciativa de se criar uma base contendo diversos componentes de software, servindo como fonte de recursos durante o processo de reutilização, não é nova. A organização, recuperação e manipulação desses componentes é que se torna um aspecto fundamental no desenvolvimento de bases de componentes, de forma que o acesso a esses recursos seja feito de maneira adequada e eficiente.

Uma base de componentes que possui, apenas, informação sobre o código fonte ou código objeto de componentes e cujo acesso é realizado através de um mecanismo de indexação simples (ex. por palavra-chave) não é, suficientemente, útil ao processo de desenvolvimento baseado em reutilização de componentes, por dois aspectos.

Primeiro, ao analisarmos um componente, a fim de estabelecer sua utilidade em uma dada aplicação, passamos por diversos estágios que vão desde a compreensão de sua função básica e observações sobre as decisões de projeto, até detalhes mais específicos de sua implementação. Note que o conhecimento sobre a análise e projeto de um componente não mais se encontra em seu código fonte, dificultando, assim, a tarefa de determinação da adequação das características do componente aos requisitos da nova aplicação.

Segundo, o acesso a componentes através de um mecanismo de indexação simples é, somente, capaz de representar componentes sobre um determinado aspecto (i.e. a palavra-chave correspondente). É necessário, porém, descrever componentes sobre diferentes aspectos, ou visões, além de ser capaz de manter uma certa correspondência entre estas diferentes visões.

Um outro fator importante na implementação de uma base de componentes diz respeito ao volume de componentes armazenados e a facilidade e eficiência para sua recuperação. Assumindo que o volume de componentes reutilizáveis de uma base tende a crescer com o tempo e que, portanto, seu acesso em disco pode se tornar

demasiadamente ineficiente, estabelece-se a necessidade de um *gerente de componentes* capaz de manter componentes referenciados em memória, decidindo o momento adequado para o processo de transferência de componentes do disco para memória e reduzindo ao máximo o número de acessos ao disco.

Assim sendo, estabelece-se como principais requisitos da base de componentes de software do ambiente CAOS:

1) capacidade de armazenagem de informações diversas sobre o componente (i.e. não apenas seu código fonte mas informações gerais sobre sua funcionalidade, detalhes de projeto, exemplos de utilização, etc);
2) mecanismos para organização de componentes em coleções que possam ser referenciadas sobre diferentes aspectos, determinando diferentes *visões* ou *contextos*, mantendo o relacionamento entre as diversas coleções, e
3) mecanismos para transferência de componentes entre o disco e a memória (i.e. necessidade de um gerente de componentes).

3.2. O Navegador

De uma maneira geral, existem duas formas para recuperação de elementos de uma base de componentes:

1) através de consultas diretas à base de componentes, utilizando-se, por exemplo, uma linguagem de consulta capaz de exprimir as características do componente procurado, e
2) através da navegação sobre a estrutura organizacional da base, à procura de componentes com características, ainda, não muito bem estabelecidas.

Esses dois métodos não são exclusivos, podendo ser utilizados conjuntamente, de forma a se complementarem. Assim sendo, a partir de uma primeira consulta tentativa, começa-se a percorrer a base à procura de componentes mais específicos ou, da mesma forma, a partir de uma primeira visualização das características

encontradas em objetos observados durante um processo de navegação, opta-se pela definição de uma consulta mais específica.

No ambiente CAOS, é prevista a implementação destes dois métodos em um mesmo elemento do ambiente, denominado o *navegador*, cujas principais funções são:

1) localizar componentes reutilizáveis similares, candidatos a uma determinada função ou atividade;
2) permitir a compreensão desses componentes, e
3) permitir a navegação entre os diversos componentes da base.

3.3. O Acoplador de Rotinas Externas

Um outro aspecto considerado relevante ao projeto do ambiente CAOS é prover facilidades de comunicação com outras linguagens de programação, permitindo a reutilização de código previamente definido em linguagens procedimentais (ex. FORTRAN ou C). Desta forma, o ambiente provê mecanismos para o encapsulamento de rotinas externas em objetos manipuláveis pelo sistema.

A ferramenta, denominada *Acoplador*, tem como principal função preparar o software externo ao sistema CAOS, integrando-o ao sistema. Como principais requisitos desta ferramenta, temos:

1) a possibilidade de acoplamento (i.e. integração) em dois níveis:
   a) *acoplamento mínimo*: descreve o protocolo de comunicação de rotinas externas, definindo seu nome e parâmetros formais e,
   b) *acoplamento estruturado*: define objetos complexos a partir de rotinas externas já acopladas e objetos disponíveis no sistema;
2) alimentação da rede de informações sobre o novo componente acoplado, para o atendimento do requisito 1) da base de componentes (i.e. armazenagem de informações diversas sobre o novo componente), e

3) determinação da classificação do novo componente segundo o esquema de facetas utilizado (ver seção 3).

3.4. O Gerente e a Linguagem de Composição de Aplicações

De acordo com Stadelmann [STAD91], existem duas possíveis maneiras de abordar o problema da composição de componentes em linguagens orientadas a objetos:

1) através da extensão de construtores da linguagem, ou

2) através de ferramentas interativas.

No ambiente CAOS, adota-se uma solução híbrida para este problema que fornece, além de uma ferramenta interativa para auxílio durante o processo de composição, denominado *gerente de composição de aplicações* (CAOMA - "Composing Application-Objects' MAnager"), uma linguagem de programação capaz de resolver questões sobre a composição de objetos, denominada CAOL ("Composing Application-Objects Language"). Esta linguagem utiliza os mecanismos tradicionais de composição de linguagens orientadas a objetos (i.e. troca de mensagem e herança) e introduz um mecanismo para verificação de tipos, em tempo de compilação, antecipando possíveis erros de execução e possibilitando a avaliação do impacto de mudanças durante a adaptação de componentes.

O modelo de objetos desta linguagem segue, essencialmente, o modelo de linguagens de programação orientadas a objetos, tal como Smalltalk [GOLD84] ou Actor [WHIT91], no que concerne os conceitos de classe, objeto e mecanismo de herança.

3.5. O Monitor

O objetivo da ferramenta de monitoração é permitir a execução de aplicações construidas no ambiente CAOS, de forma a possibilitar a observação de seu comportamento. Para isto, é preciso criar um mecanismo de controle do conjunto de instâncias de classes da aplicação criadas durante sua execução, provendo um ambiente similar ao espaço de trabalho da linguagem Actor [WHIT91].

Baseado nos resultados obtidos durante a execução da aplicação, o desenvolvedor poderá decidir entre a inclusão da nova aplicação na base de componentes do ambiente, para futura reutilização, e/ou a continuação do processo de desenvolvimento, substituindo e/ou evoluindo componentes até que um resultado satisfatório seja obtido.

4. O Protótipo do Ambiente CAOS

A partir dos requisitos do ambiente CAOS estabelecidos na seção anterior, implementou-se um primeiro protótipo do ambiente na linguagem Actor 3.0 [WHIT91], rodando sobre MS-Windows 3.0, em uma plataforma de microcomputadores 486.

O protótipo do ambiente CAOS é formado pór uma coleção de classes em Actor que implementam os conceitos necessários para o atendimento dos requisitos do ambiente. A *figura 1* mostra os principais conceitos envolvidos no protótipo do ambiente CAOS.

A característica principal do ambiente está relacionada à sua organização geral sobre a forma de hipertexto. Portanto, os conceitos de *nó* e *ligação* entre nós são o ponto de partida para compreensão dos demais elementos do ambiente. Um nó é uma unidade atômica de informação sobre um componente. Todo nó possui uma janela associada, que consiste na interface do usuário para acesso às operações

permitidas de um nó, através de cardápios de comandos e movimentos de mouse. Nós podem ser do tipo *descrição, classe, método, papel, faceta* ou *rotina*.

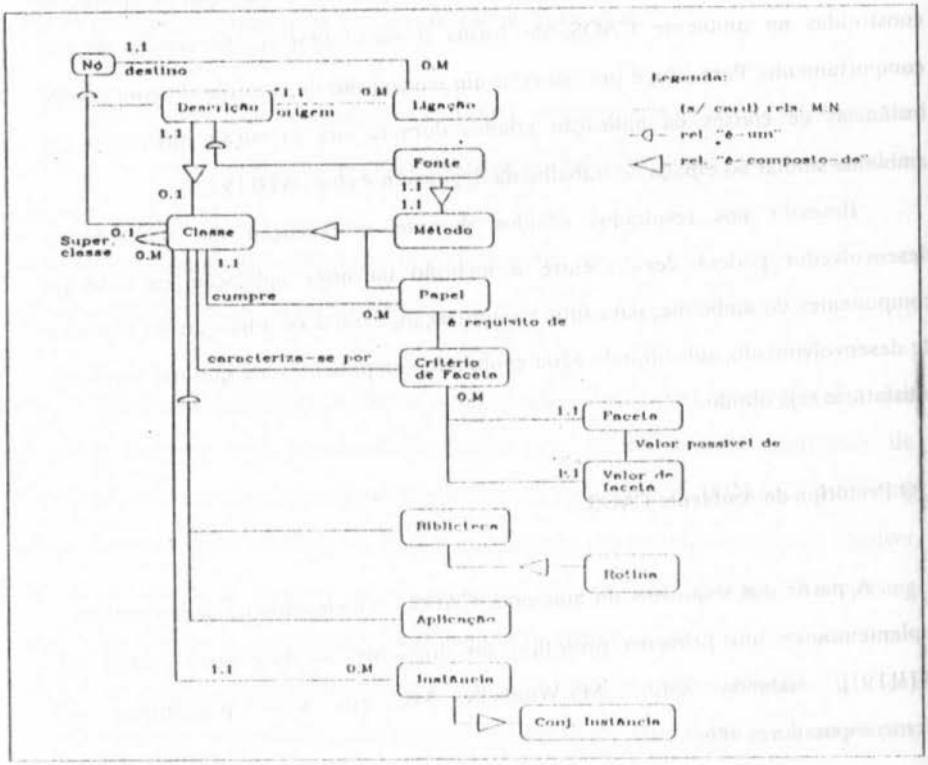

Fig. 1 - Principais conceitos envolvidos no protótipo do Ambiente CAOS[1].

Um *nó de descrição* consiste em um conjunto de palavras que descrevem informações textuais sobre um componente. Nós de descrição podem estar associados entre si através de *ligações* do tipo "palavra-nó", ou seja, determinadas palavras de um nó podem levar a outros nós de informação relacionados.

1 O diagrama usado nesta figura é uma adaptação do diagrama proposto em [COAD92]. Para manter a legibilidade da figura, alguns relacionamentos foram suprimidos.

Um nó do tipo *classe* contém informações estruturais e comportamentais sobre um componente. Um nó de classe tem sempre um nó de descrição associado que, a partir da ligação com outros nós de descrição, permite a criação de uma rede de informações textuais sobre o componente. Classes (i.e. nós do tipo classe) podem estar relacionadas entre si através de ligações do tipo "palavra-nó" de seu nó de descrição associado. Ligações deste tipo podem ser estabelecidas de forma a implementar relações de similaridade entre classes.

As informações estruturais e comportamentais de uma classe consistem em um conjunto de *métodos* e *papéis*. *Métodos* em CAOS traduzem o mesmo significado de métodos em outras linguagens de programação orientadas a objetos (LPOOs). Cada método de uma classe possui um *fonte* associado, que é uma especialização de um nó de descrição com capacidade para compilação de seu conteúdo.

Papéis correspondem à variáveis de instância em outras LPOOs e podem ser atendidos por uma classe já definida ou, ainda, por definir. O nome *papel* evidencia a possibilidade de reutilização de classes já definidas a partir do cumprimento de determinados requisitos estabelecidos para o papel. Cada papel possui um nome e uma descrição feita a partir da definição de *critérios de faceta*. Através da consulta das classes existentes, cujos critérios de facetas atendam aos requisitos do papel, é possível associar uma classe ao papel.

Classes caracterizam-se por um conjunto de *critérios de faceta* que são usados no processo de seleção de classes úteis para reutilização. A determinação de critérios de faceta é feita através da associação de valores, escolhidos a partir de uma lista de *valores possíveis* para cada faceta, a uma ou mais facetas disponíveis no sistema. Uma faceta pode estar associada a um nó de descrição, possibilitando a inclusão de descrições textuais sobre a mesma.

Facetas, no protótipo do ambiente CAOS, implementam o esquema de classificação proposto por Prieto-Díaz [PRIE85] em um ambiente orientado a objetos, para recuperação de classes. O esquema de classificação por facetas consiste em

descrever relações genéricas básicas entre elementos a partir da montagem, ou síntese, de suas classes elementares. Conforme identificado por Prieto-Díaz [PRIE91], o esquema de classificação por facetas é mais eficiente quando sua aplicação se restringe a coleções específicas, aumentando seu poder descritivo. Na medida em que o ambiente CAOS pode servir a diferentes áreas de aplicação, assume-se que a lista de facetas e de seus possíveis valores será adaptada ao tipo de componentes armazenados no sistema, havendo facilidades para a gerência de facetas.

Classes em CAOS geram classes em Actor que as implementam. O processo de geração das classes Actor correspondentes é ativado a partir de uma opção de comando, no cardápio de comandos da janela do nó principal do sistema. Esta característica permite não somente a implementação de aplicações de CAOS em Actor como em qualquer outra linguagem de programação orientada a objetos em que se deseja implementar a aplicação (ex. C++) ou, ainda, em linguagens de programação convencionais.

Bibliotecas em CAOS são formadas por um conjunto de rotinas. *Rotinas* são rotinas externas acopladas ao sistema. Bibliotecas são responsáveis pelo gerenciamento de suas rotinas, acoplando-as, removendo-as e permitindo a navegação a qualquer rotina especificada. Uma biblioteca em CAOS é, na verdade, uma especialização da classe *Classe* e considera o conjunto de rotinas acopladas como parte das informações estruturais e comportamentais da classe, juntamente com métodos e papéis. Esta facilidade permite o atendimento do requisito de acoplamento mínimo, descrito na seção 3.3. A classe *Biblioteca*, enquanto classe, também, permite a criação de novos métodos e papéis, satisfazendo, assim, o requisito de acoplamento estruturado (i.e. definição de objetos mais complexos a partir de rotinas já acopladas e objetos do sistema).

Uma *aplicação* em CAOS é, também, uma especialização da classe *Classe*, consistindo de uma classe com um único método. Uma aplicação pode ser vista como um programa em linguagens de programação convencional. Este conceito facilita a

organização de aplicações fechadas, ativadas através de um comando de execução enviado a partir do monitor do sistema, que controla o *conjunto de instâncias* criadas durante sua execução.

Finalmente, todas as classes, bibliotecas e aplicações geradas em CAOS são mantidas por um *gerente de objetos*. Este elemento mantém, ainda, o relacionamento entre as classes de CAOS e as classes de Actor que as implementam.

A *figura 2* mostra uma seção típica do Ambiente CAOS, onde a classe Compilador está em fase de composição.

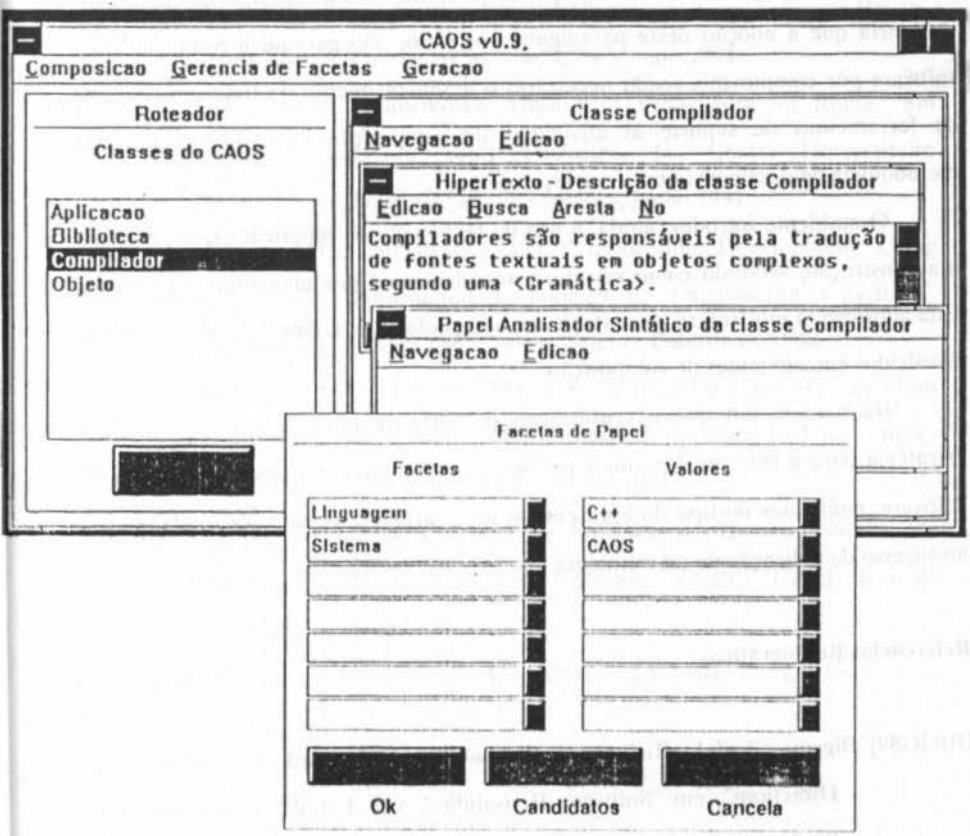

Fig. 2 - Seção típica do Ambiente CAOS

5. Conclusões

Neste artigo, apresentou-se o Ambiente CAOS, que se propõe a ser um ambiente de suporte ao desenvolvimento de aplicações baseado na reutilização por composição. O ambiente oferece mecanismos para à recuperação, compreensão, modificação e composição de componentes de uma biblioteca de software reutilizável.

O ambiente apresentado baseia-se nas facilidades incorporadas ao paradigma de orientação a objetos para reutilização de software. Nossa experiência, entretanto, evidencia que a adoção deste paradigma, por si só, não garante a reutilização de software por composição, sendo necessário o desenvolvimento de todo um conjunto de ferramentas de suporte às atividades de composição envolvidas e de uma metodologia de desenvolvimento correspondente.

O ambiente introduz, ainda, o uso da tecnologia de hipertexto como base de sua construção, servindo como estrutura organizacional dos elementos do sistema. Esta abordagem mostrou-se, perfeitamente, adequada à implementação dos conceitos envolvidos em ambientes de composição.

Na medida em que a reutilização de software passa a ser uma possível estratégia para a solução dos atuais problemas enfrentados no desenvolvimento de software, ambientes do tipo do apresentado neste artigo passam a ser fundamentais ao sucesso da aplicação de tal estratégia.

Referências Bibliográficas

[BIGG89] Biggerstaff, T.J.; Richter, C. "Reusability, Framework, Assessment, and Directions", em "Software Reusability", vol. I, (ed.) Biggerstaff, T.J.; Perlis, A.J.; Addison Wesley, 1989.

[CONK87] Conklin, J. "Hypertext: An Introduction and Survey", IEEE Computer, 20(9), setembro 1987.

[COAD92] Coad, P.; Yordon, E.; Análise Baseada em Objetos, (trad.) CTI Informática, Editora Campus, 1992.

[GOLD84] Goldberg, A. "Smalltalk-80: The Interactive Programming Environment", Addison-Wesley, Massachusetts, 1984.

[PRIE85] Prieto-Diaz, R. "A Software Classification Scheme", Ph.D.thesis, Universidade da California, 1985.

[PRIE91] Prieto-Diaz, R. "Implementing Faceted Classification for Software Reuse", Communications of the ACM, vol.34, no.5, maio 1991.

[STAD91] Stadelmann, M. "TeamWorks: Towards A Framework for Reuse", em "Object Composition" (ed.) Tsichritzis, D.; Centre Universitaire d'Informatique, Universidade de Genebra, junho 1991.

[TSIC88] Tsichritzis, D.C.; Nierstrasz, O.M. "Application Development Using Objects", em "Information Technology for Organizational Systems", Proceedings EURINFO'88, Elsevier Science Publishers, 1988.

[WERN91] Werner, C.M.L.; Souza, J.M. "An Object-Oriented Composition Environment for Scientific Applications", Computing in High Energy Physics Conference, Tsukuba City, Japão, 1991.

[WERN92] Werner, C.M.L.; "Reutilização de Software no Desenvolvimento de Software Científico", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Março 1992.

[WHIT91] The Whitewater Group "Actor User's Manual, Evanston, USA, 1991.